Attorney Docket No. 0023-0042

UNITED STATES PATENT APPLICATION

OF

Stefan DYCKERHOFF and Kong KRITAYAKIRANA

FOR

SYSTEMS AND METHODS FOR INTERFACING WITH VARIABLE SPEED STREAMS

[0001] SYSTEMS AND METHODS FOR INTERFACING WITH VARIABLE SPEED STREAMS

[0002] BACKGROUND OF THE INVENTION

[0003] Field of the Invention

[0004] The present invention relates generally to data transfer and, more particularly, to systems and methods for interfacing to multiple streams of variable speeds.

[0005] Description of Related Art

[0006] Routers receive data on physical media, such as optical fiber, analyze the data to determine its destination, and output the data on physical media in accordance with the destination. Routers were initially designed using a general purpose processor executing large software programs. As line rates and traffic volume increased, however, general purpose processors could not scale to meet the new demands. For example, as new functions, such as accounting and policing functionality, were added to the software, these routers suffered performance degradation. In some instances, the routers failed to handle traffic at line rate when the new functionality was turned on.

[0007] To meet the new demands, purpose-built routers were designed. Purpose-built routers are designed and built with components optimized for routing. They not only handled higher line rates and higher network traffic volume, but they also added functionality without compromising line rate performance.

[0008] A conventional purpose-built router may include a number of input and output ports from which it receives and transmits streams of information packets. A switching fabric may be implemented in the router to carry the packets between the ports. In a

high-performance purpose-built router, the switching fabric may transmit a large amount of information between a number of internal components.

[0009] The conventional routers are typically configured based on the speeds of the packet streams they receive. If the speed of one of the streams changes, the routers typically must be reconfigured. Reconfiguring a router is generally a complicated and time-consuming process. Also, reconfiguring in response to a change in speed of a single stream may adversely affect other streams processed by the router.

[0010] As a result, there is a need in the art for a router that can handle streams of varying speeds without requiring major reconfiguration.

[0011] SUMMARY OF THE INVENTION

[0012] Systems and methods consistent with the principles of the invention address this and other needs by providing input logic that interfaces to multiple streams of packet data with varying speeds without requiring major reconfiguration.

[0013] One aspect consistent with the principles of the invention is a system that processes packet data received in a number of incoming streams of variable speeds. The system includes an input interface, input logic, and one or more packet processors. The input interface receives the packet data and outputs the data using a first arbitration element. The input logic includes flow control logic, a memory, and a dispatch unit. The flow control logic initiates flow control on the data output by the input interface. The memory stores the data from the input interface. The dispatch unit reads the data from the memory using a second arbitration element. The packet processor(s) process the data from the dispatch unit.

[0014] In another aspect consistent with the principles of the invention, a method for processing data received in a plurality of incoming streams of variable speeds includes storing data in a memory using a first arbitration element, and reading the data from the memory using a second arbitration element.

[0015] In yet another aspect consistent with the principles of the invention, a method for performing flow control on data in a plurality of incoming streams of variable speeds includes storing data in a plurality of entries in a buffer; determining the number of entries in the buffer corresponding to each of the streams; and determining whether to initiate flow control for each of the streams based on the determined number of entries for the stream.

[0016] BRIEF DESCRIPTION OF THE DRAWINGS

[0017] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

[0018] Fig. 1 is a block diagram illustrating an exemplary routing system in which systems and methods consistent with principles of the invention may be implemented;

[0019] Fig. 2 is a detailed block diagram illustrating portions of the routing system of Fig. 1;

[0020] Figs. 3A-3C are exemplary diagrams of WAN physical links consistent with principles of the invention;

[0021] Fig. 4 is an exemplary diagram of a physical interface card (PIC) of Fig. 2;

[0022] Fig. 5 is an exemplary diagram of a flexible port concentrator (FPC) of Fig. 2;

[0023] Fig. 6 is an exemplary diagram of the input logic of Fig. 5;

[0024] Fig. 7 is an exemplary diagram of the buffer of Fig. 6;

[0025] Fig. 8 is an exemplary flowchart of initial processing of packets received by the routing system of Fig. 1 in an implementation consistent with principles of the invention;

[0026] Fig. 9 is an exemplary diagram of input logic with flow control capabilities according to implementations consistent with principles of the invention;

[0027] Fig. 10 is an exemplary diagram of flow control logic of Fig. 9 in an implementation consistent with principles of the invention; and

[0028] Fig. 11 is an exemplary flowchart of processing by flow control logic of Fig. 9 according to implementations consistent with principles of the invention.

[0029] DETAILED DESCRIPTION

[0030] The following detailed description of the invention refers to the accompanying drawings. The same reference numbers in different drawings may identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims and equivalents.

[0031] Systems and methods consistent with principles of the invention provide input logic for a multistream interface of a routing system. The input logic permits multiple input streams to interface to the routing system without requiring reconfiguration of the routing system in response to changes to the stream speed.

[0032] SYSTEM CONFIGURATION

[0033] Fig. 1 is a block diagram illustrating an exemplary routing system 100 in which systems and methods consistent with the principles of the invention may be implemented. System 100 receives a data stream from a physical link, processes the data stream to determine destination information, and transmits the data stream out on a link in accordance with the destination information. System 100 may include packet forwarding engines (PFEs) 110, a switch fabric 120, and a routing engine (RE) 130.

[0034] RE 130 performs high level management functions for system 100. For example, RE 130 communicates with other networks and systems connected to system 100 to exchange information regarding network topology. RE 130 creates routing tables based on network topology information, creates forwarding tables based on the routing tables, and forwards the forwarding tables to PFEs 110. PFEs 110 use the forwarding tables to perform route lookup for incoming packets. RE 130 also performs other general control and monitoring functions for system 100.

[0035] PFEs 110 are each connected to RE 130 and switch fabric 120. PFEs 110 receive data at ports on physical links connected to a network, such as a wide area network (WAN). Each physical link could be one of many types of transport media, such as optical fiber or Ethernet cable. The data on the physical link is formatted according to one of several protocols, such as the synchronous optical network (SONET) standard, an asynchronous transfer mode (ATM) technology, or Ethernet.

[0036] PFE 110 processes incoming data by stripping off the data link layer. PFE 110 converts header information from the remaining data into a data structure referred to as a notification. For example, in one embodiment, the data remaining after the data link

layer is stripped off is packet data. PFE 110 converts the layer 2 (L2) and layer 3 (L3) packet header information included with the packet data into a notification. PFE 110 may store the information in the notification, some control information regarding the packet, and the packet data in a series of cells. In one embodiment, the notification and the control information are stored in the first two cells of the series of cells.

[0037] PFE 110 performs a route lookup using the notification and the forwarding table from RE 130 to determine destination information. PFE 110 may also further process the notification to perform protocol-specific functions, policing, and accounting, and might even modify the notification to form a new notification.

[0038] If the destination indicates that the packet should be sent out on a physical link connected to PFE 110, then PFE 110 retrieves the cells for the packet, converts the modified notification information into header information, forms a packet using the packet data from the cells and the header information, and transmits the packet from the port associated with the physical link.

[0039] If the destination indicates that the packet should be sent to another PFE via switch fabric 120, then PFE 110 retrieves the cells for the packet, modifies the first two cells with the modified notification and new control information, if necessary, and sends the cells to the other PFE via switch fabric 120. Before transmitting the cells over switch fabric 120, PFE 110 appends a sequence number to each cell, which allows the receiving PFE to reconstruct the order of the transmitted cells. Additionally, the receiving PFE uses the notification to form a packet using the packet data from the cells, and sends the packet out on the port associated with the appropriate physical link of the receiving PFE.

[0040] In summary, RE 130, PFEs 110, and switch fabric 120 perform routing based on packet-level processing. PFEs 110 store each packet using cells while performing a route lookup using a notification, which is based on packet header information. A packet might be received on one PFE and go back out to the network on the same PFE, or be sent through switch fabric 120 to be sent out to the network on a different PFE.

[0041] Fig. 2 is a detailed block diagram illustrating portions of routing system 100. PFEs 110 connect to one another through switch fabric 120. Each of the PFEs may include one or more physical interface cards (PICs) 210 and flexible port concentrators (FPCs) 220.

[0042] PIC 210 may transmit data between a WAN physical link and FPC 220. Different PICs may be designed to handle different types of WAN physical links. For example, one of PICs 210 may be an interface for an optical link while the other PIC may be an interface for an Ethernet link.

[0043] Figs. 3A-3C are exemplary diagrams of WAN physical links consistent with principles of the invention. The examples of Figs. 3A-3C assume that the WAN link contains a total bandwidth of STS-192. In Fig. 3A, a single stream contains the entire STS-192 bandwidth. In Fig. 3B, the STS-192 bandwidth is divided among four STS-48 streams. In Fig. 3C, the STS-192 bandwidth is divided into N streams of varying bandwidth.

[0044] Returning to Fig. 2, FPCs 220 perform routing functions and handle packet transfers to and from PICs 210 and switch fabric 120. For each packet it handles, FPC 220 performs the previously-discussed route lookup function. Although Fig. 2 shows two PICs 210 connected to each of FPCs 220 and three FPCs 220 connected to switch fabric

120, in other embodiments consistent with principles of the invention there can be more or fewer PICs 210 and FPCs 220.

[0045] EXEMPLARY PIC CONFIGURATION

[0046] Fig. 4 is an exemplary diagram of a PIC 210 consistent with the principles of the invention. PIC 210 will be described in terms of the logic it may have for processing streams of packet data it receives from the WAN. It should be understood that PIC 210 may include similar or different logic for processing packet data for transmission to the WAN.

[0047] In the implementation illustrated in Fig. 4, PIC 210 includes WAN interface 410 and FPC interface controller 420. WAN interface 410 may include logic that connects to one or more particular types of physical WAN links to receive streams of packets from the WAN. For example, in one particular implementation consistent with principles of the invention, WAN interface 410 may handle 64 separate streams of packets.

[0048] WAN interface 410 may also include logic that processes packets received from the WAN . For example, WAN interface 410 may include logic that strips off the layer 1 (L1) protocol information from incoming data and forwards the remaining data, in the form of raw packets, to interface controller 420.

[0049] Interface controller 420 may include logic that determines how to allocate bandwidth to streams of packet data sent to FPC 220. Interface controller 420 can handle varying levels of granularity of stream speeds. For example, in one implementation, stream speeds are programmed to STS-3 granularity. To implement varying granularity, interface controller 420 may use an arbitration element, such as arbitration table 425, that

has entries corresponding to a particular unit of bandwidth. Arbitration elements other than tables may also be used. Arbitration table 425 might, for example, consist of 64 entries, with each entry corresponding to an STS-3's worth of bandwidth. Each entry in the table can be a 6-bit number indicating which stream is to be serviced for that arbitration slot.

[0050] For purposes of illustration, assume that arbitration table 425 allocates bandwidth among four streams, labeled as 0-3. In this case, arbitration table 425 may be configured as follows:

STREAM NUMBER
0
1
2
1
3
1
0
1
2

In this example, stream 1 is allocated much more of the bandwidth than are streams 0, 2, and 3. Interface controller 420 may read a stream number from arbitration table 425, gather data from the corresponding stream, attach a stream identifier (ID), and transmit the data to FPC 220.

[0051] Arbitration table 425 may be programmed under software control. For example, an operator may specify a speed for each of the packet streams. The software may then update arbitration table 425 based on the specified speeds. A faster stream may be given more slots and a slower stream may be given less slots in arbitration table 425.

[0052] EXEMPLARY FPC CONFIGURATION

[0053] Fig. 5 is an exemplary diagram of FPC 220 consistent with principles of the invention. FPC 220 will be described in terms of the logic it may have for processing streams of packet data it receives from PIC 210. It should be understood that FPC 220 may include similar or different logic for processing packet data for transmission to PIC 210.

[0054] In the implementation illustrated in Fig. 5, FPC 220 includes packet processor 510 and input logic 520. Packet processor 510 may process packet data received from PICs 210 and create packet data for transmission out to the WAN via PICs 210. Packet processor 510 may also process packet data received from switch fabric 120 and transmit packet data to other PFEs via switch fabric 120.

[0055] Input logic 520 may provide initial processing to packet data received from PICs 210. Fig. 6 is an exemplary diagram of input logic 520 in an implementation consistent with principles of the invention. Input logic 520 may include a buffer 610, a dispatch unit 620, and processing logic 630.

[0056] Buffer 610 may include one or more memory devices, such as one or more first-in, first-out (FIFO) static random access memories (SRAMs), that are partitioned on a per stream basis. Fig. 7 is an exemplary diagram of buffer 610 consistent with principles of the invention. Buffer 610 may include a number of memory buckets 710 equal to the number of possible packet streams. Buffer 610 may sort packet data it receives based on their stream ID and store the data in the memory bucket corresponding to the stream. The amount of memory allocated to each of the streams (i.e., the size of memory buckets 710) may be fixed regardless of the speed of the streams. For example,

in one implementation, each of memory bucket 710 may be configured to store 512 entries of 256 bits each.

[0057] Returning to Fig. 6, dispatch unit 620 may include logic that determines when and what to read from buffer 610 for processing by processing logic 630. In other words, dispatch unit 620 may determine how to allocate bandwidth to streams of packet data sent to processing logic 630. Dispatch unit 620 can handle varying levels of granularity of stream speeds. For example, in one implementation, stream speeds are programmed to STS-3 granularity.

[0058] To implement varying granularity, dispatch unit 620 may use an arbitration element, such as arbitration table 625, that has entries corresponding to a particular unit of bandwidth. Arbitration elements other than tables may also be used. Arbitration table 625 might, for example, consist of 64 entries, with each entry corresponding to an STS-3's worth of bandwidth. Each entry in the table can be a 6-bit number indicating which stream is to be serviced for that arbitration slot.

[0059] For purposes of illustration, assume that arbitration table 625 allocates bandwidth among four streams, labeled 0-3. In this case, arbitration table 625 may be configured as follows:

STREAM NUMBER
0
1
2
1
3
1
0
1
2

In this example, stream 1 is allocated much more of the bandwidth than are streams 0, 2, and 3. Dispatch unit 620 may read a stream number from arbitration table 625, read data from the memory bucket in buffer 610 that corresponds to the stream, and transmit the data to processing logic 630.

[0060] Arbitration table 625 may be programmed under software control. For example, an operator may specify a speed for each of the packet streams. The software may then update arbitration table 625 based on the specified speeds. A faster stream may be given more slots and a slower stream may be given less slots in arbitration table 625.

[0061] In an implementation consistent with principles of the invention, arbitration table 625 resembles arbitration table 425 in PIC 210. If arbitration tables 425 and 625 are synchronized, there should be no need to make the size of the memory buckets 710 in buffer 610 dependent on the stream speed because PIC 210 sends packet data for each of the streams to buffer 610 at the same rate that dispatch unit 620 reads packet data for the streams from buffer 610.

[0062] Processing logic 630 may process header information in the packet data. For example, processing logic 630 may convert the layer 2 (L2) and layer 3 (L3) packet header information (collectively, L23) into information for use by packet processor 510 (Fig. 5). Processing logic 630 may also create other information related to the packet.

[0063] INITIAL PACKET PROCESSING

[0064] Fig. 8 is an exemplary flowchart of initial processing of packets received by a system, such as routing system 100, in an implementation consistent with principles of the invention. Processing may begin with system 100 receiving streams of packet data which it analyzes and routes to its destination or a next hop toward the destination. WAN

interface 410 of PIC 210 may receive packet data belonging to multiple packet streams (act 810). WAN interface 410 may strip off the layer (L1) protocol information from the incoming packet data and forward the remaining data to FPC interface controller 420.

[0065] Interface controller 420 may determine how to allocate bandwidth to the streams of packet data it outputs. Interface controller 420 may use its arbitration table 425 to allocate bandwidth. For example, interface controller 420 may allocate a predetermined amount of bandwidth, such as an STS-3's worth of bandwidth, to each of the stream numbers in arbitration table 425. To dispatch data, interface controller 420 may read a stream number from arbitration table 425, gather packet data corresponding to the stream, attach a stream ID, and transmit the data to FPC 220 (act 820).

[0066] Buffer 610 within FPC 220 may store packet data from interface controller 420 in a memory bucket 710 according to the streams to which the packet data belong (act 830). For example, if buffer 610 receives packet data belonging to stream 0, buffer 610 stores the packet data in memory bucket 710 corresponding to stream 0. Buffer 610 may identify the stream to which particular packet data belongs based on the stream ID, or some other sort of identifying information, provided with the data.

[0067] Dispatch unit 620 may determine how to allocate bandwidth to the streams of packet data stored in buffer 610. Dispatch unit 620 may use its arbitration table 625 to allocate bandwidth. For example, dispatch unit 620 may allocate a predetermined amount of bandwidth, such as an STS-3's worth of bandwidth, to each of the stream numbers in arbitration table 625. To dispatch data, dispatch unit 620 may read a stream number from arbitration table 625, read packet data corresponding to the stream from buffer 610, and transmit the data to processing logic 630 (act 840).

[0068] Processing logic 630 may process the packet data by, for example, converting header information into a notification (act 850). For example, processing logic 630 may process layer 2 (L2) and layer 3 (L3) packet header information, and possibly other information, for processing by one or more packet processors, such as packet processors 510 (Fig. 5).

[0069] FLOW CONTROL SYSTEM

[0070] Fig. 9 is an exemplary diagram of input logic 900 with flow control capabilities according to implementations consistent with principles of the invention. Input logic 900 may be used in a system similar to system 100 (Fig. 1). In this implementation, however, FPC interface controller 420 (Fig. 4) and dispatch unit 620 (Fig. 6) may or may not have arbitration tables 425 and 625, respectively.

[0071] In the implementation shown in Fig. 9, input logic 900 includes flow control logic 910, buffer 920, dispatch unit 930, and processing logic 940. Assume for purposes of this description that buffer 920, dispatch unit 930, and processing logic 940 are configured similar to buffer 610, dispatch unit 620, and processing logic 630 described above with regard to Fig. 6.

[0072] Flow control logic 910 may provide flow control functions to input logic 900. Flow control may be necessary in cases where there is a chance that data may build up and overflow the buffer (e.g., buffer 920). One illustrative case may be where a PIC is sized for a first speed, but the bus between the PIC and the FPC supports a second faster speed. In this case, the PIC may send stream data in bursts at the second speed even though the average rate of transmission may be equal to the first speed. As a result of the burst transfers, data may build up and possibly overflow buffer 920. Flow control in this

case may include a signal sent back to the PIC to inform the PIC to stop sending data for that particular stream.

[0073] Fig. 10 is an exemplary diagram of flow control logic 910 in an implementation consistent with principles of the invention. Flow control logic 910 may include FIFO 1010, a counter 1020, and a comparator 1030. FIFO 1010 may include a memory device that stores data for any stream. FIFO 1010 temporarily buffers data from the PIC. When the data reaches the front of FIFO 1010, it is written to buffer 920.

[0074] The size of FIFO 1010 may be chosen to absorb the latency of the assertion of flow control to take effect (i.e., the time it takes for the data to stop for the stream for which flow control was asserted). The latency may be based on the round trip delay for flow control information, which may include the amount of time it takes for flow control information to reach the PIC and the PIC to react to the flow control information by generating idles. In an implementation consistent with principles of the invention, FIFO 1010 is capable of storing at least 192 entries of 128 bits each.

[0075] FIFO 1010 saves buffer space because flow control is generated out of a common buffer for all streams. If this were not the case, then the buffer space for all of the streams would have to be increased by the worst case flow control latency.

[0076] Counter 1020 may count the number of entries for each stream that are stored in FIFO 1010. In other words, counter 1020 keeps track of the stream to which each entry in FIFO 1010 belongs and counts the total number of entries in FIFO 1010 on a per stream basis.

[0077] Comparator 1030 may compare the total numbers of entries in FIFO 1010 for each of the streams to one or more watermarks per stream. The watermarks may be

controlled by software and set equal to an expected number of entries for each of the streams based, for example, on the stream's speed. The watermarks may be reprogrammed to compensate for changes in the speed of the corresponding streams.

[0078] Comparator 1030 may maintain two watermarks for each of the streams. Either or both of these watermarks may be used in implementations consistent with principles of the invention.

[0079] One watermark ("flow control (F/C) watermark") may be used by comparator 1030 when determining when to assert flow control for the corresponding stream. When the number of entries in FIFO 1010 for a stream reaches the F/C watermark, comparator 1030 may assert a flow control signal for that stream. Another watermark ("drop watermark") may be used by comparator 1030 when determining whether to drop data in the corresponding stream. When the number of entries in FIFO 1010 for a stream reaches the drop watermark, comparator 1030 may drop data in that stream.

[0080] In implementations consistent with principles of the invention, the value of the drop watermark is greater than the value of the F/C watermark. In this case, flow control logic 910 may begin flow control measures for a stream and drop data when the flow control measures fail to ease the flow of packet data in that stream. This may occur when the PIC fails to respond to flow control measures, such as the case where the PIC does not have the capabilities to perform flow control.

[0081] Using the above-described features, input logic 900 may assure that data in a particular stream does not get dropped due to a change in speed of the stream, assuming that the PIC responds to the flow control measures.

[0082] PACKET PROCESSING WITH FLOW CONTROL

[0083] Fig. 11 is an exemplary flowchart of processing by flow control logic 910 according to an implementation consistent with principles of the invention. Processing may begin with a PIC transmitting packet data in a variety of streams to flow control logic 910. Flow control logic 910 may receive the packet data and store it in FIFO 1010 in the order of their arrival (acts 1110 and 1120). Counter 1020 may monitor the reception and transmission of packet data and track the number of entries in FIFO 1010 belonging to each of the streams. Counter 1020 may count the total number of entries in FIFO 1010 on a stream-by-stream basis (act 1130).

[0084] Comparator 1030 may compare each of the total values to one or more watermarks corresponding to the stream (act 1140). For example, comparator 1030 may compare a total value to the F/C watermark and/or drop watermark for a stream. If the total value exceeds the F/C watermark, comparator 1030 may initiate flow control measures by generating a flow control signal that it sends to the PIC (act 1150). In this case, FIFO 1010 may continue to buffer packet data for the stream until the flow control measures take effect.

[0085] If the total value exceeds the drop watermark, comparator 1030 may begin to drop packet data in the stream by generating a drop signal that it sends to FIFO 1010 (act 1150). In response to the drop signal, FIFO 1010 may drop packet data in the stream from either the top or bottom of the FIFO.

[0086] In an alternate implementation, comparator 1030 sends the drop signal to buffer 920 (Fig. 9). In response to the drop signal, buffer 920 may drop packet data in the stream from either the bottom or top of the memory bucket.

[0087] When FIFO 1010 drops packet data or outputs packet data to buffer 920, counter 1020 decrements its count. Based on this decremented count, comparator 1030 may continue or discontinue asserting the flow control and/or drop signals.

[0088] OTHER IMPLEMENTATIONS

[0089] Systems have been described for initially processing packet data using arbitration tables and/or flow control measures. In other implementations consistent with principles of the invention, yet other systems may be used. In another implementation, the buffer within the input logic is partitioned based on the known speed of the streams. If one of the streams changes speed, however, the buffer is repartitioned based on the change in speed.

[0090] In yet another implementation, the buffer is partitioned with oversized memory buckets. The size of the memory buckets is chosen so that even the fastest stream will not overflow the memory bucket.

[0091] CONCLUSION

[0092] Systems and methods consistent with the principles of the invention provide input logic for a multistream interface of a routing system. The input logic permits multiple input streams to interface to the routing system without requiring reconfiguration of the routing system in response to changes to the stream speed.

[0093] When a stream is reconfigured (i.e., when the speed of the stream changes), the input logic adjusts arbitration tables and flow control and/or drop watermarks. Instead of allocating buffer space according to the speed of the stream, the processing rate is changed and the buffer space is fixed. Because the buffer space does not need to be reallocated, no other stream is affected and no data is lost.

[0094] The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention.

[0095] For example, although described in the context of a routing system, concepts consistent with the principles of the invention can be implemented in any system that processes and buffers data.

[0096] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items. Where only one item is intended, the term "one" or similar language is used. The scope of the invention is defined by the claims and their equivalents.